UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,642	05/01/2001	Luciano Lavagno	CA7012162001	6620

55497 7590 05/15/2008
BINGHAM MCCUTCHEN LLP
THREE EMBARCADERO CENTER
SAN FRANCISCO, CA 94111-4067

EXAMINER	
GUILL, RUSSELL L	

ART UNIT	PAPER NUMBER
2123	

MAIL DATE	DELIVERY MODE
05/15/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 09/847,642	**Applicant(s)** LAVAGNO ET AL.
	Examiner Russ Guill	**Art Unit** 2123

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 February 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4-22,33,34 and 36-60* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4-22,33,34 and 36-60* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 July 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to an amendment filed February 15, 2008. No claims were added or canceled. Claims 1 - 2, 4 - 22, 33 - 34 and 36 - 60 are pending. Claims 1 - 2, 4 - 22, 33 - 34 and 36 - 60 have been examined. Claims 1 - 2, 4 - 22, 33 - 34 and 36 - 60 are rejected.

2. **The Examiner would like to thank the Applicant for the well-presented amendment, which was useful in the examination process. The Examiner appreciates the effort to carefully analyze the Office Action and make appropriate arguments and amendments.**

Continued Examination Under 37 CFR 1.114

3. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on February 15, 2008, has been entered.

Response to Remarks

4. Regarding Correction of Inventorship:

   4.1. The inventorship now appears to be corrected.

5. Regarding claims 14 and 55 objected to for minor informalities:

5.1. Applicant's claim amendments overcome the objections.

6. Regarding claims 33 - 34, 36 - 40 and 41 - 54 rejected under 35 USC § 101:

6.1. Applicant's claim amendments overcome the rejections.

7. Regarding claims 1 - 2, 4 - 13, 15 - 19, 21 - 22, 33 - 34, 36 - 45, 47 - 51, 53 - 55, 57 - 58 and 60 rejected under 35 USC § 103:

7.1. Applicant's arguments have been fully considered, but are not persuasive, as discussed in the following. Accordingly, the rejections are maintained.

7.2. The Applicant argues:

7.3. Claim 1 recites at least the following limitations. Claims 9, 33, 41, and 55 also recite similar limitations.

7.3.1. *generating* a software simulation model ***by translating*** the assembler code or ***disassembling*** a binary code ***into a high level language format and by annotating*** the software simulation model with information related to estimation or determination of the performance of hardware on which the software program runs ***to capture an interaction between tasks during runtime;***

7.3.2. (emphasis added.)

7.4. According to the final Office Action, <u>Passerone</u> does not disclose or suggest the above limitation but <u>Hellestrand</u> and <u>Zivojnovic</u> do. The final Office Action further acknowledges that <u>Hellestrand</u> does not disclose the limitation of "translating the assembler code . . ."

Applicants respectfully agree with the final Office Action that Passerone does not disclose, teach, or suggest the above claimed limitations and that Hellestrand does not disclose or suggest the limitation of "translating the assembler code .. .", but Applicants respectfully disagree that Zivoinovic does.

7.4.1. The Examiner respectfully replies:

7.4.2. The Office action relies upon the combination of Passerone, Hellestrand and Zivojnovic to teach the limitation. Please refer to the rejection. However, the focus of the Applicant's argument that follows appears to be directed to Zivojnovic, and so the issue appears to be moot.

7.5. The Applicant argues:

7.6. (a) The final Office Action cites to p. 692 § IV of Zivoinovic and purports that the cited passages disclose the above claimed limitation. Applicants respectfully disagree.

7.7. (i) Zivoinovic discloses an approach to make the translation process less complex and more portable by using the compiled simulation approach which translates a target instruction into one or more host instructions, where both the target instructions and the host instructions are assembly codes. § IV, *left hand column, second and third paragraphs and right hand column, fourth paragraph*.

7.8. Nonetheless, Applicants respectfully submit that this is not the claimed limitation. Zivoinovic explicitly states that "Nowever, **compiled-simulation assumes that the code does not change during run-time.** Therefore, self-modifying programs will **force us to use a hybrid interpretive/compiled scheme.** Fortunately, self-modifying programs are rare . . . ." *§ IV, right hand column, third paragraph*. That is, the compiled-simulation approach as described in section IV of Zivojnovic only applies when the code does **not change during run-time**, and

Zivo'novic applies the interpretative simulators when the code does change during run-time. In contrast, claim 1 specifically recites "generating a software simulation model ... and by **annotating** the software simulation model . . . **to capture a dynamic interaction between tasks during runtime".** Therefore, Applicants respectfully submit that Zivojnovic's compiled-simulation approach as described in § IV does not disclose or suggest the aforementioned limitation. In fact, Zivojnovic explicitly states that the compiled-simulation approach **does not apply** to the situation where the code changes during run-time.

7.9. As such, Applicants respectfully submit that Zivojnovic may not disclose at least the aforementioned claimed limitation of "generating a software simulation model ... to capture a dynamic interaction between tasks during runtime". As such, Applicants respectfully submit that Passerone Hellestrand, Zivojnovic, and their combinations do not disclose all the limitations of these claims and thus may not be used to preclude the patentability of these claims under 35 U.S.C. § 103(a).

7.9.1. The Examiner respectfully replies:

7.9.2. While the Examiner appreciates the Applicant's argument, the Examiner respectfully disagrees, as follows. First, the added limitation, "**to capture a dynamic interaction between tasks during runtime**" appears to be an intended use, and therefore does not limit the claim.

7.9.3. Second, it was common knowledge in the art that compiled-simulation would not apply to self-modifying programs, but the amended limitation does not appear to recite or suggest a self-modifying program. A dynamic interaction of tasks during runtime does not appear to mean self-modifying code. Further, the method described in the specification does not appear to be directed to self-modifying code.

7.9.4. Finally, Passerone appears to teach the limitation of capturing a dynamic interaction between tasks during runtime, as described in the rejection below.

7.9.5. Accordingly, the rejection is maintained.

7.10. The Applicant argues:

7.11. (b) Although the final office action does not rely on Hellestrand in forming the basis for rejecting the above limitation, Applicants nonetheless respectfully submit that Figs. 3A-3B of Hellestrand illustrate the flow chart of the **static analysis** process and how the information of the **static analysis** is used. Applicants thus respectfully submit that Hellestrand does not disclose, teach, or suggest the above limitations.

7.11.1. The Examiner respectfully replies:

7.11.2. The Examiner agrees that Figs. 3A – 3B of Hellestrand appear to illustrate the flowchart of a static analysis process. This would be a problem in the case of self-modifying code, but the claims do not appear to recite or imply self-modifying code, and the method described in the specification does not appear to be directed to self-modifying code.

7.12. The Applicant argues:

7.13. As such, Applicants respectfully submit that since Zivojnovic does not disclose, teach, or suggest at least the above limitation, and since the final office action acknowledges that Passerone and Hellestrand do not disclose the same, Passerone. Hellestrand, Zivojnovic, and their combination may not be used to preclude the patentability of 1, 9, 33, 41, 55, and their respective dependent claims under 35 U.S.C. § 103(a) for at least the reasons provided in subsections (a) and (1).

7.13.1. The Examiner respectfully replies:

7.13.2. The rejections are maintained for the reasons discussed above.

7.14. The Applicant argues:

7.15. Claims 14 and 46 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Passerone as modified by Hellestrand and Zivoinovic in view of Hartoog et al, *Generation of Software Tools from Processor Descriptions for Hardware/Software Codesign,* Proceedings of the 34th Design Automation Conference, June 9-13, 1997 (hereinafter Hartoog). Applicants respectfully traverse.

7.16. As the final Office Action does not rely on Hartoog in forming the basis for rejection of the limitation in section IV-A, Applicants respectfully submit that claims 14 and 46 are believed to be allowable over Passerone Hellestrand Zivoinovic, Hartoog, and their combination for at least the foregoing reasons as presented in section IV-A above and their dependency on claim 1 and 41.

7.16.1. The Examiner respectfully replies:

7.16.2. The rejections of the independent claims are maintained as discussed above, and thus, the rejections of the dependent claims are also maintained.

7.17. The Applicant argues:

7.18. Claims 20, 52, and 59 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Passerone *as* modified by Hellestrand and Zivoinovic further in view of Suzuki et al, *Efficient Software Performance Estimation Methods for Hardware/Software Codesign,* 1996 Proceedings of the 33'd Annual Conference on Design Automation (hereinafter Suzuki). Applicants respectfully traverse.

7.19. As the final Office Action does not rely on Suzuki in forming the basis for rejection of the limitation in section IV-A, Applicants respectfully submit that claims 20, 52, and 59 are believed to be allowable over Passerone, Hellestrand, Zivojnovic, Suzuki, and their combination for at least the foregoing reasons as presented in section IV-A above and their dependency on claim 1 and 41

7.19.1. The Examiner respectfully replies:

7.19.2. The rejections of the independent claims are maintained as discussed above, and thus, the rejections of the dependent claims are also maintained.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

a. Claims 9 - 22 and 41 - 54 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

i. Regarding independent claim 9, the claim recites, "annotating the software simulation model with performance information of a hardware together with which the simulation model runs". This subject matter was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The simulation model does not appear to run with the hardware, as described in the claim. The MPEP recites (section 2163), "when filing an amendment an applicant should show support in the original disclosure for new or amended claims."

ii. Regarding independent claim 41, the claim recites, "annotating the simulation model with performance information of hardware together with which the simulation model runs". This subject matter was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The simulation model does not appear to run with the hardware, as described in the claim.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

11. Claims 1 - 2, 4 - 13, 15 - 19, 21 - 22, 33 - 34, 36 - 45, 47 - 51, 53 - 55, 57 - 58 and 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Passerone (Claudio Passerone et al.; "Fast hardware/software co-simulation for virtual prototyping and trade-off analysis", 1997, Proceedings of Design Automation Conference 1997) in view of Hellestrand (U.S. Patent Number 6,230,114), further in view of Zivojnovic (Vojin Zivojnovic, "Compiled HW/SW Co-simulation", 1996, art supplied by the Applicant on the Information Disclosure Statement dated September 10, 2001, element AP).

11.1. Regarding **claims 1 and 33:**

11.2. Passerone appears to teach:

11.2.1. Describing a design for the target machine as a network of logical entities (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.2.2. Selecting at least one of the logical entities for a software implementation (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.2.3. Implementing a source software program from the logical entities selected for the software implementation (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.2.4. generating an optimized assembler code for the software program (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; it would have been obvious that C code is compiled in order to run a simulation, and that a compiler generates optimized assembler code*);

11.2.5. Generating a software simulation model ~~by translating the assembler code or disassembling a binary code into~~ *as* a high level language format and by annotating the software simulation model with information related to estimation or determination of the performance of hardware on which the software program runs to capture a dynamic interaction between tasks during runtime (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above", and page 3, section 2.2, it would have been obvious that communication events between CFSM's were a dynamic interaction between tasks, and "The C code is used in Ptolemy as a model for both hardware and software components"*);

11.2.6. Generating a hardware and software co-simulation model using the simulation model (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above" and "The C code is used in Ptolemy as a model for both hardware and software components"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."*);

11.2.7. storing at least the hardware and software co-simulation model (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above" and "The C code is used in Ptolemy as a model for both hardware and software components"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."; it would have been obvious to the ordinary artisan to save the model in order to be able to run the model multiple times*).

11.3. Passerone does not specifically teach:

11.3.1. Performing a performance analysis using the assembler code;

11.3.2. ~~Generating a software simulation model~~ by translating the assembler code or disassembling a binary code into ~~a high level language format and by annotating the software simulation model with information related to estimation or determination of the performance of hardware on which the software program runs to capture a dynamic interaction between tasks during runtime~~.

11.4. Hellestrand appears to teach:

11.4.1. Generating an optimized assembler code for a software program (*figure 3A, elements 303, 305, 307, 309, 311; and column 25, lines 56 - 67, column 26, lines 1 - 18*).

11.4.2. Performing a performance analysis using the assembler code (*figure 3A, elements 311, 313; and column 26, lines 6 - 18*);

11.4.3. Generating a software simulation model using the assembler code (*figure 3A, elements 311, 313, 315, 317, 319, 331, 333; and column 32, lines 14 - 36*);

11.5. Zivojnovic appears to teach:

11.5.1. Generating a software simulation model by translating the assembler code or disassembling a binary code (*page 692, section IV Compiled Simulation of Programmable Architectures*);

11.6. The motivation to use the art of Hellestrand with the art of Passerone would have been the benefit recited in Hellestrand that there is an important advantage to the system – a linear block can be as short as a single instruction, and the user has the option of so analyzing the code to get instruction-by-instruction timing (*column 25, lines 27 - 33*).

11.7. The motivation to use the art of Zivojnovic with the art of Passerone would have been the benefit recited in Zivojnovic that the method offers up to three orders of magnitude faster simulation (**page 690, Abstract**).

11.8. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Hellestrand and Zivojnovic and Passerone to produce the claimed invention.

11.9. Regarding **claims 2 and 34**:

11.10. Passerone appears to teach:

11.10.1. the compiling step further comprises incorporating a description of the target machine (*page 1, right-side column, third paragraph that starts with, "It is based on using . . ."*);

11.11. Regarding **claims 4 and 36**:

11.12. Passerone appears to teach:

11.12.1. selecting at least one of the logical entities for a hardware implementation, and synthesizing a software model of the hardware implementation from the selected logical entities, wherein the hardware and software co-simulation model is generated using the software model of the hardware implementation (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above" and "The C code is used in*

Ptolemy as a model for both hardware and software components"; and page 1, right-side column, third paragraph that starts with, "It is based on . . .");

11.13. Regarding **claims 5 and 37**:

11.14. Passerone does not specifically teach:

11.14.1. the performance analysis measures an execution time of an element of the assembler code.

11.15. Hellestrand appears to teach:

11.15.1. the performance analysis measures an execution time of an element of the assembler code (*figure 3A, elements 311, 313; and column 26, lines 6 - 18*).

11.16. Regarding **claims 6 and 38**:

11.17. Passerone does not specifically teach:

11.17.1. the software program is compiled using the same compiler used to compile a production executable.

11.18. Hellestrand appears to teach:

11.18.1. the software program is compiled using the same compiler used to compile a production executable (*figure 3A, elements 331, 333*).

11.19. Regarding **claims 7 and 39**:

11.20. Passerone appears to teach:

11.20.1. performing the performance analysis comprises annotating the code with performance information (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*).

11.21. Passerone does not specifically teach:

11.21.1. performing the performance analysis comprises annotating the *assembler* code with performance information.

11.22. Hellestrand appears to teach:

11.22.1. assembler code (*figure 3A, element 311; it would have been obvious to annotate assembler code because it was old and well known to annotate code with performance information; for example, Verilog HDL includes syntax to annotate code with timing information*);

11.23. Regarding **claims 8 and 40**:

11.24. Passerone appears to teach:

11.24.1. the performance information is timing information (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*).

11.25. Regarding **claims 9 and 41**:

11.26. Passerone appears to teach:

11.26.1. obtaining a software ~~assembly~~ code module from a source code module (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.26.2. generating a software simulation model ~~by translating the assembly code module into the software simulation model~~ in a high level language format (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.26.3. annotating the software simulation model with performance information of a hardware together with which the software simulation model runs to capture a dynamic interaction between tasks during runtime (*page 3, section 2.2, first paragraph, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above", and "The C code is used in Ptolemy as a model for both hardware and software components"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."; further, it was old and well known in the art to annotate code with performance information; for example, Verilog HDL includes syntax to annotate code with timing information*);

11.26.4. storing at least the software simulation model (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above" and "The C code is used in Ptolemy as a model for both hardware and software components"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."; it would have been obvious to the ordinary artisan to save the model in order to be able to run the model multiple times*);

11.27. Passerone does not specifically teach (the missing parts are indicated in ***bold, italic, underline***):

11.27.1. ~~obtaining a software~~ assembly ~~code module from a source code module~~;

11.27.2. ~~generating a software simulation model~~ by translating the assembly code module into the software simulation model ~~in a high level language format~~.

11.27.3. wherein the software simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

11.28. Hellestrand appears to teach:

11.28.1. obtaining a software **assembly** code module from a source code module (*figure 3A, elements 319, 331, 333*);

11.29. Zivojnovic appears to teach:

11.29.1. generating a software simulation model by translating the assembly code module into a software simulation model in a high level language format (*page 692, section IV Compiled Simulation of Programmable Architectures; and page 693, figure 1 b)*);

11.29.2. wherein the software simulation model is an assembler-level software simulation model, expressed in a high-level programming language (*page 692, section IV Compiled Simulation of Programmable Architectures; and page 693, figure 1 b)*).

11.30. Regarding **claims 10 and 42**:

11.31. Passerone does not specifically teach:

11.31.1. providing a software assembly code module comprises compiling software source code to assembly.

11.32. Hellestrand appears to teach:

11.32.1. providing a software assembly code module comprises compiling software source code to assembly (*figure 3A, elements 309, 311*).

11.33. Regarding **claims 11 and 43**:

11.34. Passerone appears to teach:

11.34.1. the software code module is compiled using a compiler adapted to create code that will execute on a first machine architecture (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to* *be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; it would have been obvious that the C code was compiled*);

11.35. Passerone does not specifically teach:

11.35.1. assembly code

11.36. Hellestrand appears to teach:

11.36.1. assembly code (*figure 3A, element 311*).

11.37. Regarding **claims 12 and 44**:

11.38. Passerone appears to teach:

11.38.1. the performance information is associated with the first machine architecture (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.39. Regarding **claims 13 and 45**:

11.40. Passerone appears to teach:

11.40.1. the simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture (*page 3, section 2.2, first paragraph, the sentence, "The* *formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.41. Regarding **claims 15 and 47**:

11.42. Passerone does not specifically teach:

11.42.1. the high-level programming language comprises a C code programming language;

11.43. Zivojnovic appears to teach:

11.43.1. the high-level programming language comprises a C code programming language (*page 693, figure 1, section b*);

11.44. Regarding **claims 16 and 48**:

11.45. Passerone appears to teach:

11.45.1. the translation step further comprises gathering information from the source code module from which the assembly code module was obtained (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; it would have been obvious that in order to annotate code with timing information that source code module provided information*);

11.46. Regarding **claims 17 and 49**:

11.47. Passerone does not appear to teach:

11.47.1. information gathered comprises high-level hints about the software assembly code module.

11.48. Hellestrand appears to teach:

11.48.1. information gathered comprises high-level hints about the software assembly code module (*figure 3A, elements 313, 315*).

11.49. Regarding **claims 18 and 50**:

11.50. Passerone does not appear to teach:

11.50.1. the performance information comprises estimated performance information.

11.51. Hellestrand appears to teach:

11.51.1. the performance information comprises estimated performance information (*figure 3A, elements 313, 315*).

11.52. Regarding **claims 19 and 51**:

11.53. Passerone does not appear to teach:

11.53.1. the performance information is statically estimated.

11.54. Hellestrand appears to teach:

11.54.1. the performance information is statically estimated (*figure 3A, elements 313, 315*).

11.55. Regarding **claims 21 and 53**:

11.56. Passerone appears to teach:

11.56.1. compiling the simulation model to a simulator host program (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."*);

11.56.2. executing the simulator host program on a simulator to allow performance measurements to be taken (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."*);

11.57. Regarding **claims 22 and 54**:

11.58. Passerone appears to teach:

11.58.1. linking an already annotated module with the simulation model (*page 3, section 2.2, first paragraph, the sentences, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"; and page 1, right-side column, third paragraph that starts with, "It is based on . . ."*);

11.59. Regarding **claim 55:**

11.60. Passerone appears to teach:

11.60.1. associating performance information comprising a predicted execution delay with an element of the software module to capture a dynamic interaction between tasks during runtime (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.60.2. ~~processing the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model using the assembly-language software module, wherein the assembler-level software~~ simulation model is expressed in a high-level programming language and is used to determine a time slot (*page 3, section 2.2, first paragraph, the sentence, "The formal specification of the system to be modeled is first translated by POLIS into a network of CFSMs, and then synthesized as timing-annotated C code as described above"*);

11.61. Passerone does not specifically teach (the missing parts are indicated in ***bold, italic, underline***):

11.61.1. *receiving the assembly language software module;*

11.61.2. *parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module;*

11.61.3. *processing the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model using the assembly language software module, wherein the assembler-level software ~~simulation model is expressed in a high-level programming language and is used to determine a time slot~~;*

11.61.4. associating performance information comprising an predicted execution delay with an element of the *assembly language* software module;

11.61.5. *displaying a result of the simulation model or storing the simulation model in a tangible computer readable media;*

11.62. Hellestrand appears to teach:

11.62.1. receiving the assembly language software module (*figure 3A, element 311*);

11.62.2. parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module (*figure 3A, elements 313, 315*);

11.62.3. processing the data structure to refine the accuracy of the simulation model (*figure 3A, elements 313, 315*);

11.62.4. associating performance information comprising an predicted execution delay with an element of the assembly language software module (*figure 3A, elements 311, 313; and column 26, lines 6 - 18*);

11.62.5. displaying a result of the simulation model or storing the simulation model in a tangible computer readable media (*figure 3A, elements 331, 333*);

11.63. Zivojnovic appears to teach:

11.63.1. generating an assembler-level software simulation model using the assembly language software module, wherein the assembler-level software simulation model is expressed in a high-level programming language (*page 692, section IV Compiled Simulation of Programmable Architectures*);

11.64. Regarding **claim 57**:

11.65. Passerone does not specifically to teach:

11.65.1. the performance information comprises an execution delay value for the element of the assembly language software module.

11.66. Hellestrand appears to teach:

11.66.1. the performance information comprises an execution delay value for the element of the assembly language software module (*figure 3A, elements 311, 313, 315, 317, 303, 319*).

11.67. Regarding **claim 58:**

11.68. Passerone does not specifically to teach:

11.68.1. the performance information is a statically computed value.

11.69. Hellestrand appears to teach:

11.69.1. the performance information is a statically computed value (*figure 3A, elements 311, 313, 315, 317, 303, 319*).

11.70. Regarding **claim 60:**

11.71. Passerone does not specifically to teach:

11.71.1. processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model.

11.72. Hellestrand appears to teach:

11.72.1. processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model (*figure 3A, elements 311, 313, 315, 317, 303, 319*).

12. **Claims 14 and 46** are rejected under 35 U.S.C. 103(a) as being unpatentable over Passerone as modified by Hellestrand and Zivojnovic as applied to claims **1 – 2, 4 - 13, 15 – 19, 21 - 22, 33 – 34, 36 - 45, 47 – 51, 53 – 55, 57 – 58 and 60** above, further in view of Hartoog (Hartoog, Mark R.; Rowson, James A.; Reddy, Prakash D.; Desai, Soumya; Dunlop, Douglas D.; Harcourt, Edwin A.; Khullar, Neeti; "Generation of Software Tools from Processor Descriptions for Hardware/Software Codesign", Proceedings of the 34th Design Automation Conference, June 9 – 13 1997).

12.1. Passerone as modified by Hellestrand and Zivojnovic teaches a method for preparing software for a performance estimation as applied to claims **1 – 2, 4 - 13, 15 – 19, 21 - 22, 33 – 34, 36 - 45, 47 – 51, 53 – 55, 57 – 58 and 60** above.

12.2. Regarding **claims 14 and 46**:

12.3. Passerone as modified by Hellestrand and Zivojnovic does not specifically teach:

12.3.1. disassembling software binary code to assembly code.

12.4. Hartoog appears to teach:

12.4.1. disassembling software binary code to assembly code (*page 305, section 5*).

12.5. The motivation to use the art of Hartoog with the art of Passerone as modified by Hellestrand and Zivojnovic would have been the benefit recited in Hartoog that, using a declarative description of an instruction set makes it possible to automatically generate several useful tools such as an Instruction Set Simulator and a Compiled Instruction Set Simulator (*page 304, section 3. Tools,* *first paragraph*), which would have been recognized as important benefit to save time by the ordinary artisan.

12.6. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Hartoog with the art of Passerone as modified by Hellestrand and Zivojnovic to produce the claimed invention.

13. **Claims 20, 52 and 59** are rejected under 35 U.S.C. 103(a) as being unpatentable over Passerone as modified by Hellestrand and Zivojnovic as applied to claims **1 - 2, 4 - 13, 15 - 19, 21 - 22, 33 - 34, 36 - 45, 47 - 51, 53 - 55, 57 - 58 and 60** above, further in view of Suzuki (Suzuki, Kei; Sangiovanni-Vincentelli, Alberto: "Efficient Software Performance Estimation Methods for Hardware/Software Codesign", 1996, Proceedings of the 33rd annual conference on Design Automation).

13.1. Passerone as modified by Hellestrand and Zivojnovic teaches a method for preparing software for a performance estimation as applied to claims **1 - 2, 4 - 13, 15 - 19, 21 - 22, 33 - 34, 36 - 45, 47 - 51, 53 - 55, 57 - 58 and 60** above.

13.2. Regarding **claims 20 and 52**:

13.3. Passerone as modified by Hellestrand and Zivojnovic does not specifically teach:

13.3.1. performance information is computed dynamically at run-time, using a formula provided during the annotating step.

13.4. Suzuki appears to teach:

13.4.1. performance information is computed dynamically at run-time, using a formula provided during the annotating step (*page 5, figure 4, run-time formula to calculate C_i.max_time*).

13.4.1.1. Regarding (*page 5, figure 4, run-time calculation of C_i.max_time*); it would have been obvious to use the dynamic run-time formula as the annotation.

13.5. Regarding **claim 59**:

13.6. Passerone as modified by Hellestrand and Zivojnovic does not specifically teach:

13.6.1. performance information is a formula for dynamically computing a value.

13.7. Suzuki appears to teach:

13.7.1. performance information is a formula for dynamically computing a value (*page 5, figure 4, run-time formula to calculate C_i.max_time*).

13.8. The motivation to use the art of Suzuki with the art of Passerone as modified by Hellestrand and Zivojnovic would have been the benefit recited in Suzuki that, two methods are presented for accurate and fast estimation of software performance in embedded real-time reactive systems designed with the POLIS system (*page 1, section 1. Introduction, second paragraph that starts with, "In this paper, we . . ."*), which would have been recognized as important benefit to save time by the ordinary artisan.

13.9. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Suzuki with the art of Passerone as modified by Hellestrand and Zivojnovic to produce the claimed invention.

14. Examiner's Note: Examiner has cited particular columns and line numbers in the references applied to the claims above for the convenience of the applicant. Although the specified citations are representative of the teachings of the art and are applied to specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested from the Applicant in preparing responses, to fully consider the references in their entirety as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the Examiner. The entire reference is considered to provide disclosure relating to the claimed invention.

Conclusion

15. The prior art made of record in a prior Office action, but not relied upon is pertinent to the Applicant's disclosure:

15.1. Bradford (U.S. Patent 5,857,093) teaches a method of performance simulation for a processor which is fast and has accurate timing information: compiling source code for a target processor to assembly code, extracting timing information from the assembly code, annotating the source code with the timing information, compiling the source code for a simulation processor, running the compiled code on the simulation processor.

16. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Russ Guill whose telephone number is 571-272-7955. The examiner can normally be reached on Monday – Friday 10:00 AM – 6:30 PM.

17. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez can be reached on 571-272-3753. The fax phone number for the organization where this application or proceeding is assigned is 571-273-

8300. Any inquiry of a general nature or relating to the status of this application should be directed to the TC2100 Group Receptionist: 571-272-2100.

18. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Russ Guill

Examiner

Art Unit 2123

RG

/Paul L Rodriguez/

Supervisory Patent Examiner, Art Unit 2123